Asynchronous Handler

RunPod supports the use of asynchronous handlers, enabling efficient handling of tasks that benefit from non-blocking operations. This feature is particularly useful for tasks like processing large datasets, interacting with APIs, or handling I/O-bound operations.

Writing asynchronous Handlers

Asynchronous handlers in RunPod are written using Python's `async` and `await` syntax. Below is a sample implementation of an asynchronous generator handler. This example demonstrates how you can yield multiple outputs over time, simulating tasks such as processing data streams or generating responses incrementally.

```python
import runpod
import asyncio


async def async_generator_handler(job):
    for i in range(5):
        # Generate an asynchronous output token
        output = f"Generated async token output {i}"
        yield output

        # Simulate an asynchronous task, such as processing time for a large language model
        await asyncio.sleep(1)


# Configure and start the RunPod serverless function
runpod.serverless.start(
    {
        "handler": async_generator_handler,  # Required: Specify the async handler
        "return_aggregate_stream": True,  # Optional: Aggregate results are accessible via /run endpoint
    }
)
```

Benefits of asynchronous Handlers

- **Efficiency**: Asynchronous handlers can perform non-blocking operations, allowing for more tasks to be handled concurrently.
- **Scalability**: They are ideal for scaling applications, particularly when dealing with high-frequency requests or large-scale data processing.
- **Flexibility**: Async handlers provide the flexibility to yield results over time, suitable for streaming data and long-running tasks.

Best practices

When writing asynchronous handlers:

- Ensure proper use of `async` and `await` to avoid blocking operations.
- Consider the use of `yield` for generating multiple outputs over time.
- Test your handlers thoroughly to handle asynchronous exceptions and edge cases.

Using asynchronous handlers in your RunPod applications can significantly enhance performance and responsiveness, particularly for applications requiring real-time data processing or handling multiple requests simultaneously.

Edit this page